A dynamic programming algorithm for the local access network expansion problem

A Dynamic Programming Algorithm for the Local Access Network Expansion Problem

Olaf E. Flippo[1,2] Antoon W.J. Kolen[1,3] Arie M.C.A. Koster[1,4]
Robert L.M.J. van de Leensel[1,5]

September 16, 1996

Abstract

Technological innovations and growing consumer demand have led to a variety of design and expansion problems in telecommunication networks. In particular, local access networks have received a lot of attention, since they account for approximately 60% of total investments in communication facilities. In this paper we consider the Local Access Network Expansion Problem, in which growing demand can be satisfied by expanding cable capacities and/or installing concentrators in the network. The problem is known to be NP-hard. We present a pseudo-polynomial dynamic programming algorithm, with time complexity $\mathcal{O}(nB^2)$ and storage requirements $\mathcal{O}(nB)$, where n refers to the size of the network, and B to an upper bound on concentrator capacity. The cost structure in the network is assumed to be decomposable, but may be non-convex, non-concave, and node and edge dependent otherwise. Computational results indicate that the algorithm is very efficient and can solve medium to large scale problems to optimality within (fractions of) seconds to minutes.

1 Introduction

Over the last decade major developments have occurred in the area of telecommunications. On the one hand, there has been a continuous growth in both the number of customers using telecommunication services and the user intensity with respect to these services. On the other hand, technological innovations in, for instance, transmission and switching technologies, have led to the possibility of cost reduction, as well as to improved accessibility of existing services. In addition, deregulation of the telecommunication industry has made the market more competitive, which, combined with technological advances, has resulted in a diversity of new services, such

[1]Dept of Quantitative Economics, Maastricht University, P.O.Box 616, 6200 MD Maastricht, The Netherlands.
[2]e-mail: O.Flippo@KE.UniMaas.NL
[3]e-mail: A.Kolen@KE.UniMaas.NL
[4]e-mail: A.Koster@KE.UniMaas.NL; home page: http://www.unimaas.nl/~akoster/index.html
[5]e-mail: R.vandeLeensel@KE.UniMaas.NL

as data transmission and video applications. Due to these developments, enormous investment and cost saving opportunities in the design and expansion of computer communication networks and telecommunication systems have arised. For an overview of various problems in this field we refer to Magnanti and Wong [20] and Gavish [13].

The Operations Research literature reports fruitful combinatorial approaches to a wide range of network *design* problems. In layered networks for instance, hierarchical problems arise (see Balakrishnan, Magnanti, and Mirchandani [2], [3]), where nodes in higher levels must be connected by more reliable arcs or by arcs with larger capacity. Alternatively, connectivity requirements may be formulated in terms of the design of arc or node disjoint paths in the network (see for instance Grötschel, Monma and Stoer [15], [16]). Sancho [22] considers a hierarchical network design problem in which two predetermined nodes are to be connected via a 'backbone' path, while the remaining nodes should be connected to this path in a tree-like structure. A dynamic programing algorithm is used to find an approximate solution for the problem.

A second class of network design problems involves multi-commodity flows, in which arcs typically have a fixed charge cost structure per commodity. Hellstrand, Larsson and Migdalas [17] discuss a formulation with a quasi-integral polytope, which allows the problem to be solved by a modified pivoting scheme. A similar problem can be found in Bienstock et al. [8] and Bienstock and Günlük [9], where a cutting-plane approach is applied to tackle the problem.

Since growing demand has led to capacity problems in existing networks, a lot of research focusses on network capacity *expansion* problems. Ahuja et al. [1] study the optimal expansion of transshipment networks so as to minimize the costs of network flow, subject to a constraint on the amount of expansion costs. Chang and Gavish [10] describe a multi-period expansion problem, where future demand is known for a number of periods, and the present value of future expansion costs is to be minimized (see also Shulman and Vachani [23] and Jack, Kai and Shulman [18]).

Design and expansion problems arise frequently in Local Access Networks (LANs), which typically have a tree structure. If the number of users in the LAN is small, a star configuration in which all users are directly connected to the central component is often employed. As the number of users increases, however, costs can be reduced by introducing a more general tree structure, and by installing concentrators in the network. Together with the central component, these concentrators could form a backbone network (ring structure) to increase reliability, or they can also simply be connected to the central component via a direct line. To optimize these networks, a two-phase procedure is often proposed. In the first phase the user nodes are partitioned into regions, while in the second phase concentrators are placed in each of these regions to meet traffic demand between user nodes (see for instance Gouveia and Paixão [14], Pirkul and Nagarajan [21], and Bienstock [7]). For an overview of different capacity expansion problems in LANs we refer to Balakrishnan et al. [4] and Gavish [12].

In this paper we discuss the Local Access Network Expansion Problem (LANEP). The LAN is a tree which connects user nodes to a switching center located in the root of the tree. Each user node has a traffic demand that must be routed to the switching center. Conceptually, this can be done in two ways. Demand can either be routed via its unique path to the root of the

tree, or it can be routed to a concentrator that is located elsewhere in the tree, which, on its turn, is connected to the switching center by a dedicated line *not* belonging to the tree. Thus, projected demand can be satisfied by expanding edge (i.e. cable) capacities and/or installing new concentrators. The key issue in LANEP is to find an efficient trade-off between edge expansion and concentrator installation costs.

Balakrishnan, Magnanti and Wong [5] mention that LANEP is NP-hard; they use Lagrangian relaxation, valid inequalities and preprocessing techniques to determine (near-)optimal solutions. For the special case which only involves one uncapacitated concentrator type with a piecewise-linear and concave cost structure, they show that the design problem is solvable in polynomial time (see also Barany, Edmonds and Wolsey [6]). The same problem is also studied by Cho and Shaw [11], where a dynamic programming algorithm is described that is embedded in a column generation approach. For the special case where existing edge capacities are zero (refered to as the *design* problem), their algorithm solves the problem in $\mathcal{O}(n^2B)$ time complexity and storage space. For the expansion problem a similar approach is proposed, but to our understanding the resulting algorithm is incorrect, since it may fail to find an optimal solution for certain problem instances (cf. Section 3).

In this paper, we present a dynamic programming algorithm for the expansion problem which runs in $\mathcal{O}(nB^2)$ time and requires $\mathcal{O}(nB)$ storage space, with n refering to the number of nodes in the tree (minus 1), and B to an upper bound on concentrator capacity. Our algorithm can handle general cost structures for cable expansion and concentrator installation. These structures include non-convex and non-concave costs, which may also be node and edge dependent. The only assumption we impose is *decomposability*, i.e. total cable expansion (concentrator installation) costs are the sum of the individual expansion (installation) costs per edge (node). To the authors' knowledge, this is the first exact pseudo-polynomial time algorithm for LANEP. Computational experiments indicate that the proposed algorithm is very efficient; networks up to 50 nodes can be solved within (fractions of) seconds, whereas significantly larger instances up to 1000 nodes can be solved within seconds to minutes, depending on network structure and concentrator size B.

The remainder of this paper is organized as follows. In section 2 we give a detailed description and state a mathematical formulation of the problem. In section 3 we embed the problem into two parametrized families of subproblems, and we derive relations between the members of these families. The dynamic programming algorithm, a proof of its correctness and an illustrative example are given in section 4. Computational results are addressed in Section 5. The final remarks and issues for future research in Section 6 conclude the paper.

2 Problem Description

In LANEP a tree is given in which a number of user nodes are connected to the switching center in the root of the tree. Each user node typically represents a collection of individual users connected by an underlying network. Communication between user nodes of this and other LANs is accomplished through the switching center. Therefore, instead of using traffic

demand between pairs of nodes, we can assume that each user node has a demand which must be routed to the central switching center. This may be accomplished in two ways, viz. either by routing the demand to the switching center via its unique path in the tree, or by routing it to one of the concentrators that are (to be) installed in the network. A concentrator compresses all incoming low frequency signals (demand) into one outgoing high frequency (or optical) signal, which is then routed to the switching center. It is assumed that this outgoing signal either requires negligible capacity in the network, or is routed to the switching center via a dedicated line *not* belonging to the network. The costs of constructing such dedicated lines are included in the installation costs of the concentrators involved. In practice, a large variety of electronic devices are available to compress signals. For the problem at hand, we can simply treat them as concentrators with different capacities and operational costs.

Let $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ be the tree on which LANEP is defined, with $\mathcal{V} = \{0, \ldots, n\}$ and $\mathcal{E} = \{1, \ldots, n\}$. For the purpose of our algorithm, the only restriction that is imposed on the numbering of the nodes, is that a child has a higher number than its parent in the tree. For ease of reference, we assume that both nodes and edges in $\mathcal{T}$ are numbered in a depth-first order. This implies that edge $e \in \mathcal{E}$ equals $\{p_v, v\}$, where $v \in \mathcal{V}$ and e have the same (numerical) label, and p_v is the predecessor of v. Although these conventions may seem confusing at first sight, ambiguities will always be resolved by context. For every node $v \in \mathcal{V}$ a traffic demand (load) r_v and a real-valued cost function K_v is given, where $K_v(k_v)$ specifies the concentrator costs that are involved when a load of k_v is to be processed by a concentrator in node v. Likewise, for every edge $e \in \mathcal{E}$ a real-valued cost function L_e is given, where $L_e(\ell_e)$ specifies the cable costs that are involved when a traffic load of ℓ_e is to be transfered over edge e. Due to the generality of these cost structures, a variety of problem characteristics can be taken into account as special cases. The situation where a concentrator with capacity $\hat{b}_v$ is already operational in node v for instance, can be accounted for by setting $K_v(k_v)$ equal to the costs of installing and operating a new or supplementary concentrator if the (planned) load k_v exceeds the current capacity $\hat{b}_v$, and equal to zero otherwise. Possible demolition costs of such an existing concentrator in v could thereby be included in $K_v(k_v)$. Moreover, the situation where it is allowed to install multiple concentrators in a node can be handled by the model. Finally, the cost structures that are considered in Balakrishnan, Magnanti and Wong [5] and Cho and Shaw [11] can be accounted for. In both studies concentrators are available in different types. Every concentrator of type t has a given capacity $\hat{b}^t$, and (node dependent) fixed and variable installation costs $\hat{F}_v^t$ and $\hat{c}_v^t$ respectively.[1] This situation is recovered from our cost structure by setting $K_v(k_v)$ equal to $\min_t\{\hat{F}_v^t + k_v \cdot \hat{c}_v^t \mid \hat{b}^t \geq k_v\}$ for $k_v > 0$, and zero otherwise. The edge costs that are considered in these papers have a similar structure. For every edge $e \in \mathcal{E}$ an existing capacity b_e, and (edge dependent) fixed and variable expansion costs F_e and c_e are defined; cable expansion costs are then obtained by setting $L_e(\ell_e)$ equal to $F_e + (\ell_e - b_e) \cdot c_e$ for $\ell_e > b_e$, and zero otherwise.

For $v, w \in \mathcal{V}$, let $P(v, w)$ denote the path from v to w in $\mathcal{T}$, with $V(v, w)$ the node set and $E(v, w)$ the edge set of $P(v, w)$. We define d_v to be the number of children of node v in $\mathcal{T}$, and $D_v = \{s_v^1, \ldots, s_v^{d_v}\}$ as the set of its children, where s_v^i is the i^{th} child of v. For $v \in \mathcal{V}$ and $0 \leq i \leq d_v$ we define the subtree $T[v, i]$, which is induced by node v, its first i children $\{s_v^1, \ldots, s_v^i\}$ and all successors of these children (see Johnson and Niemi [19]). We also define

[1] On close inspection, the variable costs in Cho and Shaw [11] are assumed *in*dependent of the concentrator type t.

$V[v,i]$ and $E[v,i]$ to be the node and edge sets of $T[v,i]$ respectively. The main issue in LANEP is to decide for each node $v \in \mathcal{V}$ whether to route its demand to the switching center (i.e. to the concentrator in the root node) via its unique path in the tree, or to route it to a node $w \in \mathcal{V} \setminus \{0\}$ in which a concentrator must then be installed to transmit all incoming load to the root of $\mathcal{T}$ via a dedicated line. If the load of node v is routed to a concentrator in node w, we say that v "homes on" w (cf. Balakrishnan, Magnanti and Wong [5]). The routing of demand in the LAN is restricted to the following conditions.

1. **Single level concentration:** demand is concentrated at most once before reaching the switching center in the root of the tree;
2. **Nonbifurcated routing:** for each user node its entire demand is processed by a single concentrator (possibly at the root);
3. **Contiguity condition:** if a node v homes on a concentrator in node w, then all nodes on the path from v to w home on w.

Condition 1 reflects guidelines of network planners who, given the current relative costs of cable expansion and concentrator installation, consider multiple levels of concentration to be uneconomical. Conditions 2 and 3 are enforced to ensure operational convenience of maintenance and repair (for a detailed description of these conditions, we refer to Balakrishnan, Magnanti and Wong [5]). Due to the contiguity condition, and the fact that a concentrator is installed in the root to warrant communication between this and other LANs, a node $v \in \mathcal{V}$ cannot home on just any node; rather, let W_v be the set of nodes on which node v may home, then $W_0 = \{0\}$, and if $v \in V[s_0^i, d_{s_0^i}]$ for some $0 < i \leq d_0$, then $W_v = V[s_0^i, d_{s_0^i}] \cup \{0\}$. We will now define the decision variables which enable us to state a mathematical formulation of the problem. Let

$$
\begin{array}{lcll}
x_{uw} & = & \begin{cases} 1 & \text{if node } u \text{ homes on node } w \\ 0 & \text{otherwise} \end{cases} & (u, w \in \mathcal{V}) \\
k_v & = & \text{the load to be processed by a concentrator in node } v & (v \in \mathcal{V}) \\
\ell_e & = & \text{the load to be transfered over edge } e & (e \in \mathcal{E})
\end{array}
$$

Then LANEP reads:

$$
\begin{array}{lll}
\min & \sum_{w \in \mathcal{V}} K_w(k_w) + \sum_{e \in \mathcal{E}} L_e(\ell_e) & (1) \\
\text{s.t.} & x_{00} = 1 & (2) \\
& \sum_{w \in \mathcal{V}} x_{uw} = 1 \qquad \forall u \in \mathcal{V} & (3) \\
& x_{u'w} \geq x_{uw} \qquad \forall u, u', w \in \mathcal{V} : u' \in V(u,w) & (4) \\
& k_w = \sum_{u \in \mathcal{V}} r_u \cdot x_{uw} \qquad \forall w \in \mathcal{V} & (5) \\
& \ell_e = \sum_{u,w \in \mathcal{V} : e \in E(u,w)} r_u \cdot x_{uw} \qquad \forall e \in \mathcal{E} & (6) \\
& k_w \leq B \qquad \forall w \in \mathcal{V} & (7) \\
& x_{uw} \in \{0,1\},\ k_w \geq 0,\ \ell_e \geq 0 \qquad \forall u, w \in \mathcal{V},\ \forall e \in \mathcal{E} & (8)
\end{array}
$$

The objective function in (1) defines the total costs that follow from the network expansion program (x, k, ℓ). As can be seen from its formulation, it propagates the decomposability assumption on costs. Constraint (2) states that a concentrator is installed in the root node. Constraint (3) implies that every node homes on exactly one node, whereas (4) enforces the contiguity condition. Note that (4) contains a lot of redundancy. However, since the model is only used to prove the validity of our dynamic programming approach, efficiency in the number of constraints is not an issue here. Constraints (5) and (6) define the resulting loads on the nodes and edges respectively. In (7) we assume that for any given $w \in \mathcal{V}$, a uniform bound B exists that restricts the sum of the loads of all nodes homing on w to B. Note that this assumption can be made without loss of generality, since the sum of all the loads in the tree $\mathcal{T}$ is such a bound. On the other hand, if the running time of an algorithm depends on such a bound (like our procedure does), then it probably pays to specify a more efficient bound if possible; under the aforementioned cost structure of Cho and Shaw [11] for instance, B could be chosen equal to the maximum of all concentrator capacities $\hat{b}^t$, which may be a lot smaller than the sum of all the loads in $\mathcal{T}$. The integrality and non-negativity constraints in (8) complete the formulation. Let us denote the set of feasible solutions by $\mathcal{F}$, hence

$$\mathcal{F} = \{(x, k, \ell) \mid (x, k, \ell) \text{ satisfies } (2)\text{–}(8)\}.$$

Finally, in addition to the decomposability of costs we will adopt the second assumption that

$$\forall v \in \mathcal{V}: \; r_v \text{ is a non-negative integer less than or equal to } B$$

Following standard practice, we define the minimum over an empty set to be ∞, and the summation over an empty set to be zero. The indicator function for a logical expression A is denoted by $1_{[A]}$, which equals 1 or 0, depending on whether A evaluates to true or false.

3 Parametrizations for LANEP

In this section we introduce two parametrized families of subproblems, followed by an intuitive preview of how our dynamic programming algorithm iterates between these subproblems. In addition, a motivation is given of why the parametrizations have been defined the way they have. In order to rigorously prove the correctness of the algorithm, we need to state and demonstrate some relevant relationships between the subproblems involved. These relations are twofold. On the one hand, we prove that solutions to a given subproblem are also solutions to the subproblems of which it is composed. This *downward compatibility of solutions* will be the subject of Section 3.1. On the other hand, we will show that solutions of certain subproblems can be combined to form a solution of the encapsulating subproblem. This *upward compatibility of solutions* will be discussed in Section 3.2. Based on these results, the final relations between the various subproblems that are used by our dynamic programming algorithm, are stated and proven in Section 3.3.

Given an arbitrary subtree $T = (V, E)$ of $\mathcal{T}$ and an arbitrary solution $(x, k, \ell) \in \mathcal{F}$, the costs of subtree T for the solution (x, k, ℓ) are defined by:

$$C(x, k, \ell \mid T) \quad = \quad \sum_{w \in V} K_w(k_w) + \sum_{e \in E} L_e(\ell_e) \tag{9}$$

Note that for $w \in V$, the variable k_w also contains the load of nodes that do *not* belong to T, but that do home on w. Similarly, ℓ_e may contain load from outside (inside) T that is transfered over e to a concentrator inside (outside) T. Next, for (v, i) with $v \in \mathcal{V}$, $0 \leq i \leq d_v$ and $0 \leq s \leq B - r_v$ we define

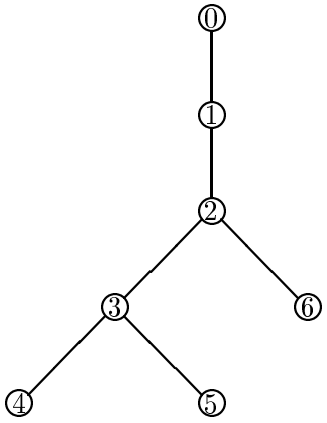
$$\begin{aligned}
g(v, i, s) \quad = \quad & \min \quad C(x, k, \ell \mid T[v, i]) && && (10)\\
& \text{s.t.} \quad x_{vw} = 0 && \forall w \notin V[v, i] && (11)\\
& \qquad \textstyle\sum_{u \notin V[v,i]} \sum_{w \in V[v,i]} r_u \cdot x_{uw} = s && && (12)\\
& \qquad (x, k, \ell) \in \mathcal{F} && && (13)
\end{aligned}$$

and for (v, i) with $v \in \mathcal{V}$, $0 \leq i \leq d_v$ and $r_v \leq r \leq B$ we define

$$\begin{aligned}
h(v, i, r) \quad = \quad & \min \quad C(x, k, \ell \mid T[v, i]) && && (14)\\
& \text{s.t.} \quad x_{vw} = 0 && \forall w \in V[v, i] && (15)\\
& \qquad \textstyle\sum_{u \in V[v,i], w \notin V[v,i]} r_u \cdot x_{uw} = r && && (16)\\
& \qquad (x, k, \ell) \in \mathcal{F} && && (17)
\end{aligned}$$

Hence, $g(v, i, s)$ represents the minimal costs of subtree $T[v, i]$ among all solutions $(x, k, \ell) \in \mathcal{F}$ for which v homes within $T[v, i]$, and the total demand of nodes *not* in $T[v, i]$ homing within $T[v, i]$ equals s. Note that by contiguity, this load s must home on the same node as node v, which explains the bound on s. Similarly, $h(v, i, r)$ represents the minimal costs of subtree $T[v, i]$ among all solutions $(x, k, \ell) \in \mathcal{F}$ for which node v does *not* home within $T[v, i]$ and the total demand of nodes in $T[v, i]$ homing outside $T[v, i]$ equals r. Again, note that this load r must home on the same node as node v.

Our dynamic programming algorithm operates in a bottom-to-top kind of fashion as follows. Suppose that in the example of Figure 1 subtree $T[2, 2]$ is under consideration for the calculation of $g(2, 2, s)$ (recall that $V[2, 2] = \{2, 3, 4, 5, 6\}$). Since $T[2, 2]$ is composed of the two encapsulated subtrees $T[6, 0]$ and $T[2, 1]$ (along with edge 6), the general idea is to obtain an optimal solution to the former by combining optimal solutions of the latter two. Since in $g(2, 2, s)$ node 2 must home within $T[2, 2]$, two situations may arise. On the one hand, it may be optimal to combine an optimal solution to $T[6, 0]$ with node 6 homing on some node in $T[6, 0]$ (necessarily node 6 in this example), with an optimal solution to $T[2, 1]$ with node 2 homing on some node not in $T[6, 0]$. In this case the former of the two optimal solutions is an optimal solution for $g(6, 0, 0)$ and the latter is an optimal solution for $g(2, 1, s)$. On the other hand it may be optimal to



v	r_v	c_v	F_v	b_v	$\hat{c}_v$	$\hat{F}_v$
0	0	–	–	–	0	0
1	4	M	M	2	M	M
2	6	1	10	4	M	M
3	6	2	8	7	M	M
4	2	M	M	21	1	3
5	5	M	M	5	1	5
6	5	M	M	5	3	10

Figure 1: Example.

combine optimal solutions of $T[6,0]$ and $T[2,1]$ with nodes 2 and 6 both homing on the same node w. In case $w \in V[2,1]$ load from $T[6,0]$ is transfered to $T[2,1]$ via edge 6. Since the resulting costs depend on the load that is transfered over edge 6, it is necessary to know how much load is actually involved. Let α denote the load over edge 6, then the solution for $g(2,2,s)$ can be obtained by combining solutions $h(6,0,\alpha)$ and $g(2,1,s+\alpha)$. In case in case $w \in V[6,0]$ load is transferred from $T[2,1]$ to $T[6,0]$ via edge 6, and the solution for $g(2,2,s)$ can be obtained by combining solutions $g(6,0,s+\alpha)$ and $h(2,1,\alpha)$. The above explains the occurrence of the parameter s in $g(v,i,s)$ and r in $h(v,i,r)$. For the calculation of $h(2,2,r)$ several cases can be distinguished in a similar manner.

So, roughly speaking, our dynamic programming algorithm proceeds as follows. While passing through the list of subtrees $T[v,i]$ two situations may arise. If $i=0$ then $T[v,i]$ consists of a single node, and computing all $g(v,i,r)$ and $h(v,i,r)$ values is a trivial exercise. On the other hand, if $i>0$ then the tree $T[v,i]$ is composed of its immediate predecessor $T[s_v^i, d_{s_v^i}]$ and its ancestor $T[v,i-1]$ (along with edge s_v^i), and solutions to the former are constructed from solutions to the latter two. Once $g(0,d_0,0)$ is obtained, the original problem is solved. Sections 3.1–3.3 validate these general ideas.

Next let us focus on the paramater s included in the first parametrization $g(v,i,s)$. Consider again the example of Figure 1. The data in this example describe the cost structure of Cho and Shaw [11] in the case of a single concentrator type (see Section 2). To be more specific, F_v and c_v denote the fixed and variable costs of capacity expansion of *edge* v, with b_v the existing cable capacity of v, and $\hat{F}_v$ and $\hat{c}_v$ denote the fixed and variable costs of concentrator installation in

node v, with r_v the demand of v. In our terminology,

$$K_v(k_v) = \begin{cases} \hat{F}_v + \hat{c}_v \cdot k_v & \text{if } k_v > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$L_e(\ell_e) = \begin{cases} F_e + c_e \cdot (\ell_e - b_e) & \text{if } \ell_e > b_e \\ 0 & \text{otherwise} \end{cases}$$

Finally, M specifies a large number. The costs of any solution in which node 1 homes on the root node exceeds $F_1 = M$, since $r_1 = 4 > 2 = b_1$, implying that capacity expansion of edge 1 is required in that case. The same observation holds for node 1 homing on node $1, 2, 3, 5$ or 6, since in each of these cases either an expensive concentrator must be installed or an expensive edge capacity must be expanded. Hence, node 1 should home on node 4, and by contiguity so should nodes $2, 3$ and 4. Note that the capacity of edge 3 must be expanded for this, since $r_1 + r_2 = 10 > 7 = b_3$. Also note that the load on edge 4 equals 16 in that case, which leaves a slack of 5 on that edge. The latter observation also implies that it is impossible to home both 5 and 6 on 4 without incurring costs $F_4 = M$. So, there are three possible solutions which cost less than M, viz.

(i). $x_{00} = 1$, $x_{14} = x_{24} = x_{34} = x_{44} = 1$, $x_{55} = 1$, $x_{64} = 1$, with costs 60;

(ii). $x_{00} = 1$, $x_{14} = x_{24} = x_{34} = x_{44} = 1$, $x_{54} = 1$, $x_{66} = 1$, with costs 65;

(iii). $x_{00} = 1$, $x_{14} = x_{24} = x_{34} = x_{44} = 1$, $x_{55} = 1$, $x_{66} = 1$, with costs 70;

At a certain stage during the algorithm, the subtree $T[2, 2]$ is under consideration. Compare the following (partial) solutions for $T = T[2, 2]$.

(iv). $x_{24} = x_{34} = x_{44} = x_{55} = x_{64} = 1$ with partial costs 48, and

(v). $x_{24} = x_{34} = x_{44} = x_{54} = x_{66} = 1$ with partial costs 47.

Note that both solutions have a total load of nodes in $T[2, 2]$ homing on node 4 equal to 19. What makes the former solution expensive, is the fact that the fixed costs of expanding edge 3 are incurred, whereas in the latter solution they are not. However, this is exactly the reason why the "currently non-optimal" solution in (iv) is better suited to receive the additional load from node 1 at a later stage of the algorithm. Hence, in the absence of a parameter s that contains the load that may enter the current subtree during a later stage of the algorithm, solutions as in (iv) may be eliminated from further consideration. The example shows that, unfortunately, the overall optimal solution may then be eliminated as well.[2]

Let us conclude these general observations by noting that $g(v, i, s)$ and $h(v, i, r)$ may not have feasible solutions, since constraints (12) and (16) may be impossible to satisfy. Finally, the

[2]This is exactly the reason why the algorithm by Cho and Shaw [11] fails to compute an optimal solution for certain problem instances.

following lemmas, which are implied by the contiguity condition, will prove to be helpful in the subsequent analysis. We confine to stating the proof of the first lemma, as the remaining proofs are similar.

Lemma 3.1 *Consider (v,i) with $v \in \mathcal{V}$ and $1 \leq i \leq d_v$. Let $(x,k,\ell) \in \mathcal{F}$ be a solution for which $x_{vw} = 0$ for all $w \notin V[s_v^i, d_{s_v^i}]$. Then $x_{uw} = 0$ for all $u,w \in \mathcal{V}$ with $v \in V(u,w)$ and $w \notin V[s_v^i, d_{s_v^i}]$.*

Proof. Let $u,w \in \mathcal{V}$ be such that $v \in V(u,w)$ and $w \notin V[s_v^i, d_{s_v^i}]$. Suppose that $x_{uw} = 1$. Since $v \in V(u,w)$ it follows by (4) and (8) that $x_{vw} = 1$, a clear contradiction. ■

Lemma 3.2 *Consider (v,i) with $v \in \mathcal{V}$ and $1 \leq i \leq d_v$. Let $(x,k,\ell) \in \mathcal{F}$ be a solution for which $x_{vw} = 0$ for all $w \notin V[v, i-1]$. Then $x_{uw} = 0$ for all $u,w \in \mathcal{V}$ with $v \in V(u,w)$ and $w \notin V[v,i-1]$.*

Lemma 3.3 *Consider (v,i) with $v \in \mathcal{V}$ and $0 \leq i \leq d_v$. Let $(x,k,\ell) \in \mathcal{F}$ be a solution for which $x_{vw} = 0$ for all $w \notin V[v,i]$. Then $x_{uw} = 0$ for all $u,w \in \mathcal{V}$ with $v \in V(u,w)$ and $w \notin V[v,i]$.*

Lemma 3.4 *Consider (v,i) with $v \in \mathcal{V}$ and $0 \leq i \leq d_v$. Let $(x,k,\ell) \in \mathcal{F}$ be a solution for which $x_{vw} = 0$ for all $w \in V[v,i]$. Then $x_{uw} = 0$ for all $u,w \in \mathcal{V}$ with $v \in V(u,w)$ and $w \in V[v,i]$.*

3.1 Downward compatibility of solutions

Consider a pair (v,i) with $v \in \mathcal{V}$ and $1 \leq i \leq d_v$. Below we will show that a feasible solution for $g(v,i,s)$ is feasible for either both $g(s_v^i, d_{s_v^i}, s+\alpha)$ and $h(v,i-1,\alpha)$ for some α with $r_v \leq \alpha \leq B - s - r_{s_v^i}$, feasible for both $h(s_v^i, d_{s_v^i}, \alpha)$ and $g(v,i-1,s+\alpha)$ for some α with $r_{s_v^i} \leq \alpha \leq B - s - r_v$, or feasible for both $g(s_v^i, d_{s_v^i}, 0)$ and $g(v,i-1,s)$. To not withdraw the reader's attention from the main results, the proof of Lemma 3.5 is listed at the end of the subsection. The proofs of the other lemmas are analogous and therefore left to the reader.

Lemma 3.5 *Consider (v,i) with $v \in \mathcal{V}$ and $1 \leq i \leq d_v$. Let $(x,k,\ell) \in \mathcal{F}$ be a feasible solution for $g(v,i,s)$ with $x_{vw} = 0$ for all $w \notin V[s_v^i, d_{s_v^i}]$. Let α be such that $\ell_{s_v^i} = s + \alpha$. Then*

(i). (x,k,ℓ) is feasible for both $g(s_v^i, d_{s_v^i}, s+\alpha)$ and $h(v,i-1,\alpha)$;

(ii). $C(x,k,\ell \mid T[v,i]) = C(x,k,\ell \mid T[s_v^i, d_{s_v^i}]) + C(x,k,\ell \mid T[v,i-1]) + L_{s_v^i}(s+\alpha)$;

(iii). $r_v \leq \alpha \leq B - s - r_{s_v^i}$.

Lemma 3.6 *Consider (v,i) with $v \in \mathcal{V}$ and $1 \leq i \leq d_v$. Let $(x,k,\ell) \in \mathcal{F}$ be a feasible solution for $g(v,i,s)$ with $x_{s_v^i w} = 0$ for all $w \notin V[v,i-1]$. Let α be such that $\ell_{s_v^i} = \alpha$. Then*

(i). (x,k,ℓ) *is feasible for both* $h(s_v^i, d_{s_v^i}, \alpha)$ *and* $g(v, i-1, s+\alpha)$*;*

(ii). $C(x,k,\ell \mid T[v,i]) = C(x,k,\ell \mid T[s_v^i, d_{s_v^i}]) + C(x,k,\ell \mid T[v,i-1]) + L_{s_v^i}(\alpha)$*;*

(iii). $r_{s_v^i} \leq \alpha \leq B - s - r_v$*.*

Lemma 3.7 *Consider* (v,i) *with* $v \in \mathcal{V}$ *and* $1 \leq i \leq d_v$*. Let* $(x,k,\ell) \in \mathcal{F}$ *be a feasible solution for* $g(v,i,s)$ *with* $x_{vw} = 0$ *for all* $w \notin V[v,i-1]$ *and* $x_{s_v^i w} = 0$ *for all* $w \notin V[s_v^i, d_{s_v^i}]$*. Then*

(i). (x,k,ℓ) *is feasible for both* $g(s_v^i, d_{s_v^i}, 0)$ *and* $g(v,i-1,s)$*;*

(ii). $C(x,k,\ell \mid T[v,i]) = C(x,k,\ell \mid T[s_v^i, d_{s_v^i}]) + C(x,k,\ell \mid T[v,i-1]) + L_{s_v^i}(0)$*.*

The following lemmas indicate that a feasible solution for $h(v,i,r)$ is a feasible solution to either both $h(s_v^i, d_{s_v^i}, \alpha)$ and $h(v,i-1,r-\alpha)$ for some α with $r_{s_v^i} \leq \alpha \leq r - r_v$, or feasible for both $g(s_v^i, d_{s_v^i}, 0)$ and $h(v,i-1,r)$.

Lemma 3.8 *Consider* (v,i) *with* $v \in \mathcal{V}$ *and* $1 \leq i \leq d_v$*. Let* $(x,k,\ell) \in \mathcal{F}$ *be a feasible solution for* $h(v,i,r)$ *with* $x_{s_v^i w} = 0$ *for all* $w \in V[s_v^i, d_{s_v^i}]$*. Let* α *be such that* $\ell_{s_v^i} = \alpha$*. Then*

(i). (x,k,ℓ) *is feasible for both* $h(s_v^i, d_{s_v^i}, \alpha)$ *and* $h(v,i-1,r-\alpha)$*;*

(ii). $C(x,k,\ell \mid T[v,i]) = C(x,k,\ell \mid T[s_v^i, d_{s_v^i}]) + C(x,k,\ell \mid T[v,i-1]) + L_{s_v^i}(\alpha)$*;*

(iii). $r_{s_v^i} \leq \alpha \leq r - r_v$*.*

Lemma 3.9 *Consider* (v,i) *with* $v \in \mathcal{V}$ *and* $1 \leq i \leq d_v$*. Let* $(x,k,\ell) \in \mathcal{F}$ *be a feasible solution for* $h(v,i,r)$ *with* $x_{s_v^i w} = 0$ *for all* $w \notin V[s_v^i, d_{s_v^i}]$*. Then*

(i). (x,k,ℓ) *is feasible for both* $g(s_v^i, d_{s_v^i}, 0)$ *and* $h(v,i-1,r)$*;*

(ii). $C(x,k,\ell \mid T[v,i]) = C(x,k,\ell \mid T[s_v^i, d_{s_v^i}]) + C(x,k,\ell \mid T[v,i-1]) + L_{s_v^i}(0)$*.*

Proof. (of Lemma 3.5).

(i). For $u \in V[s_v^i, d_{s_v^i}]$ and $w \notin V[s_v^i, d_{s_v^i}]$ it follows directly from Lemma 3.1 that $x_{uw} = 0$. Moreover, since $s + \alpha = \ell_{s_v^i}$ it follows that

$$\begin{aligned} s+\alpha &= \textstyle\sum_{u,w \in \mathcal{V}: s_v^i \in E(u,w)} r_u \cdot x_{uw} \\ &= \textstyle\sum_{u \notin V[s_v^i, d_{s_v^i}], w \in V[s_v^i, d_{s_v^i}]} r_u \cdot x_{uw} + \sum_{u \in V[s_v^i, d_{s_v^i}], w \notin V[s_v^i, d_{s_v^i}]} r_u \cdot x_{uw} \\ &= \textstyle\sum_{u \notin V[s_v^i, d_{s_v^i}], w \in V[s_v^i, d_{s_v^i}]} r_u \cdot x_{uw} \end{aligned}$$

Consequently, (x, k, ℓ) is feasible for $g(s_v^i, d_{s_v^i}, s+\alpha)$. To show the feasibility of $h(v, i-1, \alpha)$, note that $x_{vw} = 0$ follows immediately from the condition in the lemma. Next,

$$\begin{aligned}
\sum_{\substack{u \in V[v,i-1] \\ w \notin V[v,i-1]}} r_u \cdot x_{uw} &= \sum_{\substack{u \in V[v,i-1] \\ w \notin V[v,i]}} r_u \cdot x_{uw} + \sum_{\substack{u \in V[v,i-1] \\ w \in V[s_v^i, d_{s_v^i}]}} r_u \cdot x_{uw} \\
&= 0 + \sum_{\substack{u \notin V[s_v^i, d_{s_v^i}] \\ w \in V[s_v^i, d_{s_v^i}]}} r_u \cdot x_{uw} - \sum_{\substack{u \notin V[v,i] \\ w \in V[s_v^i, d_{s_v^i}]}} r_u \cdot x_{uw} \\
&= (\ell_{s_v^i} - \sum_{\substack{u \in V[s_v^i, d_{s_v^i}] \\ w \notin V[s_v^i, d_{s_v^i}]}} r_u \cdot x_{uw}) \\
&\quad -(\sum_{\substack{u \notin V[v,i] \\ w \in V[v,i]}} r_u \cdot x_{uw} - \sum_{\substack{u \notin V[v,i] \\ w \in V[v,i-1]}} r_u \cdot x_{uw}) \\
&= (s + \alpha - 0) - (s - 0) \\
&= \alpha
\end{aligned}$$

where the second and fourth equality hold by Lemma 3.1.

(ii). Follows directly from $\ell_{s_v^i} = s + \alpha$ and the decomposability of the costs.

(iii). Let $\tilde{w} \in V[s_v^i, d_{s_v^i}]$ be the node for which $x_{v\tilde{w}} = 1$. Then

$$r_v = r_v \cdot x_{v\tilde{w}} \leq \textstyle\sum_{u \in V[v,i-1], w \notin V[v,i-1]} r_u \cdot x_{uw} = \alpha$$

By contiguity, $x_{s_v^i \tilde{w}} = 1$, hence (5) and (7) yield

$$B \geq \textstyle\sum_{u \in \mathcal{V}} r_u \cdot x_{u\tilde{w}} = \sum_{u \notin V[s_v^i, d_{s_v^i}]} r_u \cdot x_{u\tilde{w}} + \sum_{u \in V[s_v^i, d_{s_v^i}]} r_u \cdot x_{u\tilde{w}} \geq s + \alpha + r_{s_v^i}$$

This completes the proof. ■

3.2 Upward compatibility of solutions

Consider a pair (v, i) with $v \in \mathcal{V}$ and $1 \leq i \leq d_v$. Below we will show that some feasible solutions for some problems on $T[v, i]$, $T[v, i-1]$ and $T[s_v^i, d_{s_v^i}]$ can be combined to obtain solutions which are simultaneously feasible on all three subtrees. Once again the proof of the first lemma is listed at the end of the subsection; the other lemmas can be proven similarly.

Lemma 3.10 *Consider (v, i) with $v \in \mathcal{V}$ and $1 \leq i \leq d_v$. Let*

- *(x^1, k^1, ℓ^1) be a feasible solution for $g(s_v^i, d_{s_v^i}, s + \alpha)$,*
- *(x^2, k^2, ℓ^2) be a feasible solution for $h(v, i-1, \alpha)$, and*

- (x^3, k^3, ℓ^3) *be a feasible solution for* $g(v, i, s)$.

Let $w^* \in V[s^i_v, d_{s^i_v}]$ *be such that* $x^1_{s^i_v w^*} = 1$. *Since the value of* $h(v, i-1, \alpha)$ *does not depend on the homing node of* v *(which is not in* $T[v, i-1]$*), assume w.l.o.g. that* $x^2_{vw^*} = 1$. *Define the composite solution* (x^4, k^4, ℓ^4) *by*

$$x^4_{uw} = \begin{cases} x^1_{uw} & \text{if } u \in V[s^i_v, d_{s^i_v}] \\ x^2_{uw} & \text{if } u \in V[v, i-1] \\ x^3_{uw} & \text{if } u \notin V[v, i], w \notin V[v, i] \\ 0 & \text{if } u \notin V[v, i], w \in V[v, i] \setminus \{w^*\} \\ \sum_{u' \in V[v,i]} x^3_{uu'} & \text{if } u \notin V[v, i], w = w^* \end{cases}$$

and (k^4, ℓ^4) *according to (5)–(6) for* $x = x^4$. *Then* (x^4, k^4, ℓ^4) *is a feasible solution for* $g(s^i_v, d_{s^i_v}, s + \alpha)$, $h(v, i-1, \alpha)$ *and* $g(v, i, s)$, *with*

$$\begin{aligned} C(x^4, k^4, \ell^4 \mid T[s^i_v, d_{s^i_v}]) &= C(x^1, k^1, \ell^1 \mid T[s^i_v, d_{s^i_v}]) \\ C(x^4, k^4, \ell^4 \mid T[v, i-1]) &= C(x^2, k^2, \ell^2 \mid T[v, i-1]) \\ C(x^4, k^4, \ell^4 \mid T[v, i]) &= C(x^1, k^1, \ell^1 \mid T[s^i_v, d_{s^i_v}]) + C(x^2, k^2, \ell^2 \mid T[v, i-1]) + L_{s^i_v}(s + \alpha) \end{aligned}$$

Lemma 3.11 *Consider* (v, i) *with* $v \in \mathcal{V}$ *and* $1 \le i \le d_v$. *Let*

- (x^1, k^1, ℓ^1) *be a feasible solution for* $h(s^i_v, d_{s^i_v}, \alpha)$,
- (x^2, k^2, ℓ^2) *be a feasible solution for* $g(v, i-1, s + \alpha)$, *and*
- (x^3, k^3, ℓ^3) *be a feasible solution for* $g(v, i, s)$.

Let $w^* \in V[v, i-1]$ *be such that* $x^2_{vw^*} = 1$. *W.l.o.g. assume that* $x^1_{s^i_v w^*} = 1$. *Define the composite solution* (x^4, k^4, ℓ^4) *as in Lemma 3.10 and* (k^4, ℓ^4) *according to (5)–(6) for* $x = x^4$. *Then* (x^4, k^4, ℓ^4) *is a feasible solution for* $h(s^i_v, d_{s^i_v}, \alpha)$ *and* $g(v, i-1, s + \alpha)$ *and* $g(v, i, s)$, *with*

$$\begin{aligned} C(x^4, k^4, \ell^4 \mid T[s^i_v, d_{s^i_v}]) &= C(x^1, k^1, \ell^1 \mid T[s^i_v, d_{s^i_v}]) \\ C(x^4, k^4, \ell^4 \mid T[v, i-1]) &= C(x^2, k^2, \ell^2 \mid T[v, i-1]) \\ C(x^4, k^4, \ell^4 \mid T[v, i]) &= C(x^1, k^1, \ell^1 \mid T[s^i_v, d_{s^i_v}]) + C(x^2, k^2, \ell^2 \mid T[v, i-1]) + L_{s^i_v}(\alpha) \end{aligned}$$

Lemma 3.12 *Consider* (v, i) *with* $v \in \mathcal{V}$ *and* $1 \le i \le d_v$. *Let*

- (x^1, k^1, ℓ^1) *be a feasible solution for* $g(s^i_v, d_{s^i_v}, 0)$,

- (x^2, k^2, ℓ^2) *be a feasible solution for* $g(v, i-1, s)$, *and*
- (x^3, k^3, ℓ^3) *be a feasible solution for* $g(v, i, s)$.

Let $w^* \in V[v, i-1]$ *be such that* $x^2_{vw^*} = 1$. *Define the composite solution* (x^4, k^4, ℓ^4) *as in Lemma 3.10 and* (k^4, ℓ^4) *according to (5)–(6) for* $x = x^4$. *Then* (x^4, k^4, ℓ^4) *is a feasible solution for* $g(s^i_v, d_{s^i_v}, 0)$ *and* $g(v, i-1, s)$ *and* $g(v, i, s)$, *for which the same cost relations hold as in Lemma 3.11 with* $\alpha = 0$.

Lemma 3.13 *Consider* (v, i) *with* $v \in \mathcal{V}$ *and* $1 \le i \le d_v$. *Let*

- (x^1, k^1, ℓ^1) *be a feasible solution for* $h(s^i_v, d_{s^i_v}, \alpha)$,
- (x^2, k^2, ℓ^2) *be a feasible solution for* $h(v, i-1, r-\alpha)$, *and*
- (x^3, k^3, ℓ^3) *be a feasible solution for* $h(v, i, r)$.

Let $w^* \notin V[v, i]$ *be such that* $x^3_{vw^*} = 1$. *W.l.o.g. assume that* $x^1_{s^i_v w^*} = x^2_{vw^*} = 1$. *Define the composite solution* (x^4, k^4, ℓ^4) *by*

$$x^4_{uw} = \begin{cases} x^1_{uw} & \text{if } u \in V[s^i_v, d_{s^i_v}] \\ x^2_{uw} & \text{if } u \in V[v, i-1] \\ x^3_{uw} & \text{if } u \notin V[v, i] \end{cases}$$

and (k^4, ℓ^4) *according to (5)–(6) for* $x = x^4$. *Then* (x^4, k^4, ℓ^4) *is a feasible solution for* $h(s^i_v, d_{s^i_v}, \alpha)$ *and* $h(v, i-1, r-\alpha)$ *and* $h(v, i, r)$, *for which the same cost relations hold as in Lemma 3.11.*

Lemma 3.14 *Consider* (v, i) *with* $v \in \mathcal{V}$ *and* $1 \le i \le d_v$. *Let*

- (x^1, k^1, ℓ^1) *be a feasible solution for* $g(s^i_v, d_{s^i_v}, 0)$,
- (x^2, k^2, ℓ^2) *be a feasible solution for* $h(v, i-1, r)$, *and*
- (x^3, k^3, ℓ^3) *be a feasible solution for* $h(v, i, r)$.

Let $w^* \notin V[v, i]$ *be such that* $x^3_{vw^*} = 1$. *W.l.o.g. assume that* $x^2_{vw^*} = 1$. *Define the composite solution* (x^4, k^4, ℓ^4) *as in Lemma 3.13 and* (k^4, ℓ^4) *according to (5)–(6) for* $x = x^4$. *Then* (x^4, k^4, ℓ^4) *is a feasible solution for* $g(s^i_v, d_{s^i_v}, 0)$ *and* $h(v, i-1, r)$ *and* $h(v, i, r)$, *for which the same cost relations hold as in Lemma 3.11 with* $\alpha = 0$.

Proof. (of Lemma 3.10)

To show feasibility, we check the individual constraints.

ad (2) For $v \neq 0$, $x^4_{00} = x^3_{00} = 1$; for $v = 0$, $x^4_{00} = x^2_{00} = 1$;

ad (3) For $u \in V[v,i]$ the result follows immediately from the feasibility of the individual solutions. For $u \notin V[v,i]$:

$$\begin{aligned} \textstyle\sum_{w\in\mathcal{V}} x^4_{uw} &= \textstyle\sum_{w\notin V[v,i]} x^4_{uw} + \sum_{w\in V[v,i]\setminus\{w^*\}} x^4_{uw} + x^4_{uw^*} \\ &= \textstyle\sum_{w\notin V[v,i]} x^3_{uw} + 0 + \sum_{w\in V[v,i]} x^3_{uw} = 1 \end{aligned}$$

ad (4) Consider a triple (u,u'',w) with $u'' \in V(u,w)$. If u and u'' are both in $V[s^i_v, d_{s^i_v}]$, both in $V[v,i-1]$ or neither in $V[v,i]$ then $x^4_{u''w} \geq x^4_{uw}$ follows directly from the contiguity of the individual solutions. On the other hand, if u and u'' are located in different subtrees, we can distinguish six cases.

Suppose $u \notin V[v,i]$ and $u'' \in V[s^i_v, d_{s^i_v}]$. From $u'' \in V(u,w)$ it follows that $w \in V[s^i_v, d_{s^i_v}]$. If $w \neq w^*$, $x^4_{uw} = 0$ and the result follows immediately. If $w = w^*$, then from $u'' \in V(u,w)$ and $u'' \in V[s^i_v, d_{s^i_v}]$ it also follows that $u'' \in V(s^i_v, w)$. Hence, $x^4_{u''w} = x^1_{u''w} \geq x^1_{s^i_v w} = 1$, and the result follows. The remaining five cases can be shown similarly.

ad (5)–(6) These constraints are satisfied by definition.

ad (7) The result immediately follows once we have shown that for all $w \in \mathcal{V}$: $k^4_w = k^j_w$ for $j = 1,2,3$ depending on whether $w \in V[s^i_v, d_{s^i_v}]$, $w \in V[v,i-1]$ or $w \notin V[v,i]$; the result then follows from the feasibility of k^j_w in (7) ($j = 1,2,3$). First consider the case $w = w^*$. Then

$$\begin{aligned} k^4_w &= \textstyle\sum_{u\in\mathcal{V}} r_u \cdot x^4_{uw} \\ &= \textstyle\sum_{u\in V[s^i_v, d_{s^i_v}]} r_u \cdot x^1_{uw} + \sum_{u\in V[v,i-1]} r_u \cdot x^2_{uw} + \sum_{u\notin V[v,i], u'\in V[v,i]} r_u \cdot x^3_{uu'} \\ &= \textstyle\sum_{u\in V[s^i_v, d_{s^i_v}]} r_u \cdot x^1_{uw} + \alpha + s \\ &= k^1_w \end{aligned}$$

Next, consider the case $w \neq w^*$. If $w \in V[s^i_v, d_{s^i_v}]$, then

$$\begin{aligned} k^4_w &= \textstyle\sum_{u\in\mathcal{V}} r_u \cdot x^4_{uw} \\ &= \textstyle\sum_{u\in V[s^i_v, d_{s^i_v}]} r_u \cdot x^1_{uw} + \sum_{u\in V[v,i-1]} r_u \cdot x^2_{uw} + \sum_{u\notin V[v,i]} r_u \cdot x^4_{uw} \\ &= \textstyle\sum_{u\in V[s^i_v, d_{s^i_v}]} r_u \cdot x^1_{uw} + 0 + 0 \\ &= \textstyle\sum_{u\in V[s^i_v, d_{s^i_v}]} r_u \cdot x^1_{uw} + \sum_{u\in V[v,i-1]} r_u \cdot x^1_{uw} + \sum_{u\notin V[v,i]} r_u \cdot x^1_{uw} \\ &= \textstyle\sum_{u\in\mathcal{V}} r_u \cdot x^1_{uw} = k^1_w \end{aligned}$$

where the third and fourth equality follow from the contiguity property and the definition of x^4. For $w \in V[v,i-1]$ and $w \notin V[v,i]$ similar results can be obtained.

ad (11) If $w \notin V[s^i_v, d_{s^i_v}]$ then $x^4_{s^i_v w} = x^1_{s^i_v w} = 0$. If $w \notin V[v,i]$ then $x^4_{vw} = x^2_{vw} = 0$.

ad (12) It holds that

$$\begin{aligned}\textstyle\sum_{u\notin V[s_v^i,d_{s_v^i}],w\in V[s_v^i,d_{s_v^i}]} r_u\cdot x^4_{uw} &= \textstyle\sum_{u\in V[v,i-1],w\in V[s_v^i,d_{s_v^i}]} r_u\cdot x^2_{uw}\\ &\quad+\textstyle\sum_{u\notin V[v,i],w\in V[s_v^i,d_{s_v^i}]} r_u\cdot x^4_{uw}\\ &= \textstyle\sum_{u\in V[v,i-1]} r_u\cdot x^2_{uw^*}+\sum_{u\notin V[v,i]} r_u\cdot x^4_{uw^*}\\ &= \alpha+\textstyle\sum_{u\notin V[v,i],u'\in V[v,i]} r_u\cdot x^3_{uu'}\\ &= \alpha+s\end{aligned}$$

Moreover,

$$\textstyle\sum_{u\notin V[v,i],w\in V[v,i]} r_u\cdot x^4_{uw} = \sum_{u\notin V[v,i]} r_u\cdot x^4_{uw^*}=\sum_{u\notin V[v,i],u'\in V[v,i]} r_u\cdot x^3_{uu'}=s$$

ad (15) If $w\in V[v,i-1]$ then $x^4_{vw}=x^2_{vw}=0$.

ad (16) $\sum_{u\in V[v,i-1],w\notin V[v,i-1]} r_u\cdot x^4_{uw}=\sum_{u\in V[v,i-1],w\notin V[v,i-1]} r_u\cdot x^2_{uw}=r$.

This completes the feasibility part of the proof. In order to prove equivalence of the costs, note that we have shown in the above that $k^4_w=k^j_w$ with $j=1,2,3$ depending on the location of w. Next we show that for all $e\in\mathcal{E}$: $\ell^4_e=\ell^j_e$, where $j=1,2,3$ depending on whether $e\in E[s_v^i,d_{s_v^i}]\cup\{s_v^i\}$, $e\in E[v,i-1]$ or $e\notin E[v,i]$. Consider the case where $e\in E[s_v^i,d_{s_v^i}]\cup\{s_v^i\}$. Note that, if $u,w\in\mathcal{V}$ are such that $e\in E(u,w)$ and $u\notin V[s_v^i,d_{s_v^i}]$, then $w\in V[s_v^i,d_{s_v^i}]$. Hence,

$$\begin{aligned}\ell^4_e &= \sum_{\substack{u,w\in\mathcal{V}:\\ e\in E(u,w)}} r_u\cdot x^4_{uw}=\sum_{\substack{u\in V[s_v^i,d_{s_v^i}],w\in\mathcal{V}:\\ e\in E(u,w)}} r_u\cdot x^1_{uw}\\ &\quad+\sum_{\substack{u\in V[v,i-1],w\in V[s_v^i,d_{s_v^i}]:\\ e\in E(u,w)}} r_u\cdot x^2_{uw}+\sum_{\substack{u\notin V[v,i],w\in V[s_v^i,d_{s_v^i}]:\\ e\in E(u,w)}} r_u\cdot x^4_{uw}\\ &= \sum_{\substack{u\in V[s_v^i,d_{s_v^i}],w\in\mathcal{V}:\\ e\in E(u,w)}} r_u\cdot x^1_{uw}\\ &\quad+\Big[\textstyle\sum_{u\in V[v,i-1]} r_u\cdot x^2_{uw^*}+\sum_{u\notin V[v,i],u'\in V[v,i]} r_u\cdot x^3_{uu'}\Big]\cdot 1_{[e\in E(v,w^*)]}\\ &= \sum_{\substack{u\in V[s_v^i,d_{s_v^i}],w\in\mathcal{V}:\\ e\in E(u,w)}} r_u\cdot x^1_{uw}+[\alpha+s]\cdot 1_{[e\in E(v,w^*)]}\\ &= \sum_{\substack{u\in V[s_v^i,d_{s_v^i}],w\in\mathcal{V}:\\ e\in E(u,w)}} r_u\cdot x^1_{uw}+\Big[\textstyle\sum_{u\notin V[s_v^i,d_{s_v^i}]} r_u\cdot x^1_{uw^*}\Big]\cdot 1_{[e\in E(v,w^*)]}\\ &= \sum_{\substack{u\in V[s_v^i,d_{s_v^i}],w\in\mathcal{V}:\\ e\in E(u,w)}} r_u\cdot x^1_{uw}+\sum_{\substack{u\notin V[s_v^i,d_{s_v^i}],w\in\mathcal{V}:\\ e\in E(u,w)}} r_u\cdot x^1_{uw}=\ell^1_e\end{aligned}$$

In case $e \in E[v, i-1]$ or $e \notin E[v, i]$ similar results hold.

As a consequence, the first two cost statements follow trivially. Furthermore,

$$C(x^4, k^4, \ell^4 \mid T[v,i]) = C(x^1, k^1, \ell^1 \mid T[s_v^i, d_{s_v^i}]) + C(x^2, k^2, \ell^2 \mid T[v, i-1]) + L_{s_v^i}(\ell^4_{s_v^i})$$

Moreover, from ad (12) it follows that $\ell^4_{s_v^i} = s + \alpha$. This completes the proof. ■

3.3 Relations between family members

Finally, we derive the recursive relations on which our dynamic programming algorithm is based.

Proposition 3.1 *Consider (v, i) with $v \in \mathcal{V}$ and $i = 0$. If $g(v, i, s) < \infty$ then*

$$g(v, i, s) = K_v(r_v + s) \tag{18}$$

Proof. Since $V[v, 0] = \{v\}$, it follows from (11), (3) and (8) that $x_{vv} = 1$. Furthermore,

$$\begin{aligned} k_v = \textstyle\sum_{u \in \mathcal{V}} r_u \cdot x_{uv} &= \textstyle\sum_{u \in V[v,i]} r_u \cdot x_{uv} + \sum_{u \notin V[v,i]} r_u \cdot x_{uv} \\ &= r_v \cdot x_{vv} + \textstyle\sum_{u \notin V[v,i], w \in V[v,i]} r_u \cdot x_{uw} \\ &= r_v + s \end{aligned}$$

The objective function in (10) thus amounts to $C(x, k, \ell \mid T[v, i]) = K_v(r + s)$. ■

Proposition 3.2 *Consider (v, i) with $v \in \mathcal{V}$ and $i = 0$. If $h(v, i, r) < \infty$ then $v \neq 0$, $r = r_v$, $r \leq B - \min_{u \notin V[v,i] \,:\, \{u,v\} \in \mathcal{E}} \{r_u\}$ and*

$$h(v, i, r) = K_v(0) \tag{19}$$

Proof. From (2) and (15) it follows directly that $v \neq 0$. Moreover, $V[v, 0] = \{v\}$ and (16) imply that $r = r_v$, and since the load r must home on a node w outside $T[v, i]$, by contiguity the first node on the path from v to w homes on w. Hence, $r \leq B - \min_{u \notin V[v,i] \,:\, \{u,v\} \in \mathcal{E}} \{r_u\}$. Finally, the objective function in (14) amounts to $C(x, k, \ell \mid T[v, i]) = K_v(0)$. ■

Proposition 3.3 *Consider (v, i) with $v \in \mathcal{V}$ and $1 \leq i \leq d_v$. Define*

$$A_\alpha \;=\; g(s_v^i, d_{s_v^i}, s + \alpha) + h(v, i-1, \alpha) + L_{s_v^i}(s + \alpha) \quad (r_v \leq \alpha \leq B - s - r_{s_v^i}) \tag{20}$$

$$B_\alpha = h(s_v^i, d_{s_v^i}, \alpha) + g(v, i-1, s+\alpha) + L_{s_v^i}(\alpha) \quad (r_{s_v^i} \leq \alpha \leq B - s - r_v) \tag{21}$$

$$C = g(s_v^i, d_{s_v^i}, 0) + g(v, i-1, s) + L_{s_v^i}(0) \tag{22}$$

$$D = \min\left[\min_{r_v \leq \alpha \leq B-s-r_{s_v^i}}\{A_\alpha\}, \min_{r_{s_v^i} \leq \alpha \leq r-r_v}\{B_\alpha\}, C\right] \tag{23}$$

Then $g(v,i,s) \geq D$. *Moreover, if* $g(v,i,s) < \infty$ *then* $g(v,i,s) = D$.

Proof. Let $(x,k,\ell) \in \mathcal{F}$ be an optimal solution for $g(v,i,s)$ (if such a solution does not exist then $g(v,i,s) = \infty \geq D$). By definition of $g(v,i,s)$, v homes in $T[v,i]$. If v homes in $T[s_v^i, d_{s_v^i}]$ then Lemma 3.5 implies for $\ell_{s_v^i} = s + \alpha$ that $r_v \leq \alpha \leq B - s - r_{s_v^i}$ and that (x,k,ℓ) is feasible for both $g(s_v^i, d_{s_v^i}, s+\alpha)$ and $h(v, i-1, \alpha)$. Furthermore,

$$\begin{aligned} g(v,i,s) &= C(x,k,\ell \mid T[v,i]) \\ &= C(x,k,\ell \mid T[s_v^i, d_{s_v^i}]) + C(x,k,\ell \mid T[v,i-1]) + L_{s_v^i}(s+\alpha) \\ &\geq A_\alpha \end{aligned}$$

For both v and s_v^i homing in $T[v,i-1]$ or for v homing in $T[v,i-1]$ and s_v^i homing in $T[s_v^i, d_{s_v^i}]$ it can be proven in an analogous manner that $g(v,i,s) \geq B_\alpha$ and $g(v,i,s) \geq C$, respectively. This establishes the fact that $g(v,i,s) \geq D$.

In order to prove the second part of the proposition, let $g(v,i,s) < \infty$. From the aforementioned result it follows that $D < \infty$. Suppose $D = A_{\alpha^*}$ for some α^* with $r_v \leq \alpha^* \leq B - s - r_{s_v^i}$. Let (x^1,k^1,ℓ^1) be an optimal solution to $g(s_v^i, d_{s_v^i}, s+\alpha^*)$, (x^2,k^2,ℓ^2) be an optimal solution to $h(v,i-1,\alpha^*)$ (the existence of both solutions is implied by the fact that $D = A_{\alpha^*} < \infty$), and (x^3,k^3,ℓ^3) be a feasible solution to $g(v,i,s)$ (for which the existence is guaranteed by $g(v,i,s) < \infty$). Applying Lemma 3.10 yields a feasible solution (x^4,k^4,ℓ^4), which is simultaneously optimal for $g(s_v^i, d_{s_v^i}, s+\alpha^*)$ and $h(v,i-1,\alpha^*)$, and feasible for $g(v,i,s)$. Consequently,

$$\begin{aligned} g(v,i,s) &\leq C(x^4,k^4,\ell^4 \mid T[v,i]) \\ &= C(x^1,k^1,\ell^1 \mid T[s_v^i, d_{s_v^i}]) + C(x^2,k^2,\ell^2 \mid T[v,i-1]) + L_{s_v^i}(s+\alpha^*) \\ &= g(s_v^i, d_{s_v^i}, s+\alpha^*) + h(v,i-1,\alpha^*) + L_{s_v^i}(s+\alpha^*) = A_{\alpha^*} = D \end{aligned}$$

If $D = B_{\alpha^*}$ for some α^* with $r_{s_v^i} \leq \alpha^* \leq r - r_v$, or $D = C$, it can be proven in a similar manner that $g(v,i,s) \leq D$. This establishes the result. ■

Proposition 3.4 *Consider* (v,i) *with* $v \in \mathcal{V}$ *and* $1 \leq i \leq d_v$. *Define*

$$E_\alpha = h(s_v^i, d_{s_v^i}, \alpha) + h(v, i-1, r-\alpha) + L_{s_v^i}(\alpha) \quad (r_{s_v^i} \leq \alpha \leq r - r_v) \tag{24}$$

$$F = g(s_v^i, d_{s_v^i}, 0) + h(v, i-1, r) + L_{s_v^i}(0) \tag{25}$$

$$G = \min\left[\min_{r_{s_v^i} \leq \alpha \leq r-r_v}\{E_\alpha\}, F\right] \tag{26}$$

Then $h(v,i,r) \geq G$. Moreover, if $h(v,i,r) < \infty$ then $h(v,i,r) = G$.

Proof. Analogous. ■

4 An $\mathcal{O}(nB^2)$ Dynamic Programming Algorithm for LANEP

The relationships between the subproblems derived in the preceding section give rise to the following dynamic programming algorithm. Recall that $g(v,i,s)$ is only defined for (v,i) with $v \in \mathcal{V}$, $0 \leq i \leq d_v$ and $0 \leq s \leq B - r_v$, whereas $h(v,i,r)$ is defined on the same (v,i) pairs with $r_v \leq r \leq B$.

DYNAMIC PROGRAMMING ALGORITHM FOR LANEP

---

forall (v,i,s) with $v \in \mathcal{V}$, $0 \leq i \leq d_v$ and $0 \leq s \leq B - r_v$ **do**
    $g(v,i,s) = \infty$; /* *initialization g* */
forall (v,i,r) with $v \in \mathcal{V}$, $0 \leq i \leq d_v$ and $r_v \leq r \leq B$ **do**
    $h(v,i,r) = \infty$; /* *initialization h* */
forall $v = n$ **downto** 0 **do begin**
    forall s with $0 \leq s \leq B - r_v$ **do**
        $g(v,0,s) = K_v(r_v + s)$;
    if $(v \neq 0)$ **then**
        $h(v,0,r_v) = K_v(0)$;
    forall $i = 1$ **to** d_v **do begin**
        forall s with $0 \leq s \leq B - r_v$ **do**
            $g(v,i,s) = D$ with D defined as in (20)–(23);
        forall r with $r_v \leq r \leq B$ **do**
            $h(v,i,r) = G$ with G defined as in (24)–(26);
    end;
end;

Unfortunately, the correctness of the algorithm does not immediately follow from the results in Section 3.3, since Proposition 3.3–3.4 do not exclude the possibility that $D < g(v,i,s) = \infty$ or $G < h(v,i,r) = \infty$. Indeed, this situation may occur, since we ignore constraint (12) and (16) during computations. In the forthcoming analysis we will distinguish between the values of $g(v,i,s)$ and $h(v,i,r)$ as defined in Section 3 on the one hand, and the ones that are computed

by the aforementioned algorithm on the other hand, by temporarily providing the latter with a superscript "c" (of "computed"). Roughly speaking, we prove that the computed values $g^c(v,i,s)$ and $h^c(v,i,r)$ may differ from the true values $g(v,i,s)$ and $h(v,i,r)$ only if these problems are infeasible, which is sufficient to prove the main result of this paper (summarized in Theorem 4.2).

Theorem 4.1 *Consider (v,i) with $v \in \mathcal{V}$ and $0 \leq i \leq d_v$. Let r and s be such that $r_v \leq r \leq B$ and $0 \leq s \leq B - r_v$. Then the following statements hold*

(i). If $g(v,i,s) < \infty$ then $g^c(v,i,s) = g(v,i,s)$;

(ii). If $r \leq B - \min_{u \notin V[v,i]\,:\,\{u,v\} \in \mathcal{E}} \{r_u\}$ then $h^c(v,i,r) = h(v,i,r)$;

Proof. First consider $i = 0$. If $g(v,i,s) < \infty$ then $g^c(v,i,s) = g(v,i,s)$ follows from (18) and the definition of $g^c(v,i,s)$ in the algorithm. If $r \leq B - \min_{u \notin V[v,i]\,:\,\{u,v\} \in \mathcal{E}} \{r_u\}$ then Proposition 3.2 states that $h(v,i,r) = K_v(0)$ if $r = r_v$, $v \neq 0$, and $h(v,i,r) = \infty$ if $r = r_v$ or $v = 0$. Hence, $h^c(v,i,r) = h(v,i,r)$ follows from (19) and the definition of $h^c(v,i,r)$ in the algorithm.

To complete the proof we use induction on the pairs (v,i) in the order as described by the algorithm. For $i = 0$ the results have just been established in the previous paragraph. Now suppose that the results hold for all pairs up to and including the predecessor of a certain pair (v,i). If $i = 0$ then the previous paragraph again validates the relevant statements for (v,i). So, suppose $i > 0$. If $g(v,i,s) < \infty$ then by Proposition 3.3 we have $g(v,i,s) = D$. If the minimum in (23) is attained for $A_{\tilde{\alpha}}$ for some $\tilde{\alpha}$ with $r_v \leq \tilde{\alpha} \leq B - s - r_{s_v^i}$, then

$$\begin{aligned} g(v,i,s) &= g(s_v^i, d_{s_v^i}, s + \tilde{\alpha}) + h(v, i-1, \tilde{\alpha}) + L_{s_v^i}(s + \tilde{\alpha}) \\ &= g^c(s_v^i, d_{s_v^i}, s + \tilde{\alpha}) + h^c(v, i-1, \tilde{\alpha}) + L_{s_v^i}(s + \tilde{\alpha}) \\ &\geq g^c(v,i,s) \end{aligned}$$

where the second equality follows from the induction hypothesis, since $g(s_v^i, d_{s_v^i}, s + \tilde{\alpha}) < \infty$ and $\tilde{\alpha} \leq B - s - r_{s_v^i} \leq B - r_{s_v^i} \leq B - \min_{u \notin V[v,i]\,:\,\{u,v\} \in \mathcal{E}} \{r_u\}$. If the minimum in (23) is attained for $B_{\tilde{\alpha}}$ for some $\tilde{\alpha}$ or for C, then $g(v,i,s) \geq g^c(v,i,s)$ follows similarly. Next we will prove the reverse inequality.

The abovementioned yields $g^c(v,i,s) \leq g(v,i,s) < \infty$. If the minimum for $g^c(v,i,s)$ is attained by 20, then

$$\begin{aligned} g^c(v,i,s) &= g^c(s_v^i, d_{s_v^i}, s + \tilde{\alpha}) + h^c(v, i-1, \tilde{\alpha}) + L_{s_v^i}(s + \tilde{\alpha}) \\ &= g(s_v^i, d_{s_v^i}, s + \tilde{\alpha}) + h(v, i-1, \tilde{\alpha}) + L_{s_v^i}(s + \tilde{\alpha}) \\ &\geq g(v,i,s) \end{aligned}$$

where the second equality is explained as follows. Firstly, since the minimum is attained by 20, $g^c(s_v^i, d_{s_v^i}, s + \tilde{\alpha})$ is actually computed. From the algorithm it then follows that $s + \tilde{\alpha} \leq B - r_{s_v^i}$

which implies that $\tilde{\alpha} \leq B - s - r_{s_v^i} \leq B - \min_{u \notin V[v,i-1]\,:\,\{u,v\} \in \mathcal{E}} \{r_u\}$. By the induction hypothesis we can therefore conclude that $h^c(v, i-1, \tilde{\alpha}) = h(v, i-1, \tilde{\alpha})$. Secondly, since $g(v,i,s) < \infty$, $h(v, i-1, \tilde{\alpha}) < \infty$, $s \leq B - \tilde{\alpha} - r_{s_v^i}$ and $\tilde{\alpha} \geq r_v$ we can construct a feasible solution (x^3, k^3, ℓ^3) for $g^c(s_v^i, d_{s_v^i}, s + \tilde{\alpha})$ using feasible solutions (x^1, k^1, ℓ^1) and (x^2, k^2, ℓ^2) for $g(v,i,s)$ and $h(v, i-1, \tilde{\alpha})$, respectively, in the following way:

$$x^3_{uw} \;=\; \begin{cases} x^1_{uw} & \text{if } u \notin V[v,i], w \notin V[v,i] \\ 0 & \text{if } u \notin V[v,i], w \in V[v,i] \backslash \{s_v^i\} \\ \sum_{u' \in V[v,i]} x^1_{uu'} & \text{if } u \notin V[v,i], w = s_v^i \\ x^2_{uw} & \text{if } u \in V[v,i-1], w \in V[v,i-1] \\ 0 & \text{if } u \in V[v,i-1], w \notin (V[v,i-1] \cup \{s_v^i\}) \\ \sum_{u' \notin V[v,i-1]} x^2_{uu'} & \text{if } u \in V[v,i-1], w = s_v^i \\ 1 & \text{if } u \in V[s_v^i, d_{s_v^i}], w = u \\ 0 & \text{if } u \in V[s_v^i, d_{s_v^i}], w \neq u \end{cases}$$

and (k^3, ℓ^3) according to (5)–(6) for $x = x^3$. Because this solution is feasible for $g(s_v^i, d_{s_v^i}, s + \tilde{\alpha})$ it follows from the induction hypothesis that $g^c(s_v^i, d_{s_v^i}, s + \tilde{\alpha}) = g(s_v^i, d_{s_v^i}, s + \tilde{\alpha})$. This justifies the second equality.

If the minimum for $g^c(v,i,s)$ is attained by 21 for some $\tilde{\alpha}$ or by 22, then $g^c(v,i,s) \geq g(v,i,s)$ follows similarly. As a result, $g(v,i,s) < \infty$ implies $g^c(v,i,s) = g(v,i,s)$. For $h(v,i,r)$, the result follows in a similar way. ■

Theorem 4.2 *Suppose $K_v(k_v)$ can be computed in $\mathcal{O}(m)$ time for every $k_v \in [r_v, B] \cap \mathbb{N}$ and $v \in \mathcal{V}$, and $L_e(\ell_e)$ can be computed in $\mathcal{O}(p)$ time for every $\ell_e \in [0, B] \cap \mathbb{N}$ and $e \in \mathcal{E}$, where m and p are parameters depending on problem size. Furthermore, let each of these computations require $\mathcal{O}(nB)$ storage space. Then the aforementioned dynamic programming algorithm finds an optimal solution in $\mathcal{O}(n(m+p)B + nB^2)$ time and $\mathcal{O}(nB)$ storage space. Under mild conditions on the cost structure (which are satisfied in the real-life applications of Balakrishnan, Magnanti and Wong [5] and Cho and Shaw [11]), it follows that $\mathcal{O}(m) = \mathcal{O}(B)$ and $\mathcal{O}(p) = \mathcal{O}(1)$, implying an overall time complexity of $\mathcal{O}(nB^2)$.*

Proof. Correctness follows directly from $g^c(0, d_0, 0) = g(0, d_0, 0)$ (cf. Theorem 4.1). As for time and space complexity, all coefficients $K_v(k_v)$ can be calculated in $\mathcal{O}(nmB)$ and all coefficients $L_e(\ell_e)$ in $\mathcal{O}(npB)$. Storage requirements for these coefficients is $\mathcal{O}(nB)$. Since the number of (v,i)-pairs we consider is $\mathcal{O}(n)$, it follows that all remaining computations can be done in $\mathcal{O}(nB^2)$. Obviously, the storage requirement for all g and h coefficients equals $\mathcal{O}(nB)$.

In practical situations it is reasonable to assume that $\mathcal{O}(m) = \mathcal{O}(B)$. For instance, in the cost structure described by Cho and Shaw [11] (cf. Section 2), the variable concentrator costs $\hat{c}_v$ do not depend on the concentrator type t. So, if two concentrators t_1 and t_2 have the same capacity $\hat{b}^{t_1} = \hat{b}^{t_2}$, and if $\hat{F}_v^{t_1} \leq \hat{F}_v^{t_2}$, then t_1 will never be more expensive to use than t_2. In other words,

for every node $v \in \mathcal{V}$ there will be at most one non-dominated concentrator per load figure k_v, implying that $\min_t\{\hat{F}_v^t + \hat{c}_v k_v \mid \hat{b}^t \geq k_v\}$ can be computed in $\mathcal{O}(m) = \mathcal{O}(B)$ time. Similarly, in Balakrishnan, Magnanti and Wong [5], the concentrator cost structure is represented by a piecewise-linear, concave function, with breakpoints occurring only at integer-valued arguments. Since such a function is described as the point-wise minimum of at most B affine functions, it follows again that $\mathcal{O}(m) = \mathcal{O}(B)$. ■

Now we have established the correctness of the procedure, we will drop the superscript "c" henceforth. Note that the algorithm in its current form only determines the optimal solution *value*, rather then an optimal *solution*. However, from Lemmas 3.10–3.14 and Propositions 3.1–3.4 it follows that an optimal solution can be recovered in the traditional way by keeping track of the "argmin" (instead of just the "min") in the evaluations of g and h coefficients, and by constructing the solution afterwards using backward recursion. As an illustration we applied the algorithm to the problem in Figure 1, with $M = 100$ for variable, and $M = 1000$ for fixed cost figures and $B = 40$. The computational results are summarized in Table 1.[3]

$g(0,0,s)$	$= 0$	$0 \leq s \leq 40$	$h(2,1,17)$	$= 21$	
$g(1,0,s)$	$= 1400 + 100s$	$0 \leq s \leq 36$	$h(2,1,19)$	$= 20$	
$h(1,0,4)$	$= 0$		$g(6,0,0)$	$= 25 + 3s$	$0 \leq s \leq 35$
$g(2,0,s)$	$= 1600 + 100s$	$0 \leq s \leq 34$	$h(6,0,5)$	$= 0$	
$h(2,0,6)$	$= 0$		$g(2,2,0)$	$= 47$	
$g(3,0,s)$	$= 1600 + 100s$	$0 \leq s \leq 34$	$g(2,2,1)$	$= 48$	
$h(3,0,6)$	$= 0$		$g(2,2,2)$	$= 54$	
$g(4,0,s)$	$= 5 + s$	$0 \leq s \leq 38$	$g(2,2,3)$	$= 57$	
$h(4,0,2)$	$= 0$		$g(2,2,4)$	$= 60$	
$g(3,1,s)$	$= 11 + s$	$0 \leq s \leq 15$	$g(2,2,s)$	$= 58 + 3s$	$5 \leq s \leq 9$
$g(3,1,s)$	≥ 1127	$16 \leq s \leq 34$	$g(2,2,s)$	≥ 1188	$10 \leq s \leq 34$
$h(3,1,6)$	$= 5$		$h(2,2,6)$	$= 41$	
$h(3,1,8)$	$= 0$		$h(2,2,11)$	$= 16$	
$g(5,0,s)$	$= 10 + s$	$0 \leq s \leq 35$	$h(2,2,12)$	$= 40$	
$h(5,0,5)$	$= 0$		$h(2,2,14)$	$= 45$	
$g(3,2,s)$	$= 16 + s$	$0 \leq s \leq 10$	$h(2,2,17)$	$= 15$	
$g(3,2,s)$	$= 21 + s$	$11 \leq s \leq 15$	$h(2,2,19)$	$= 20$	
$g(3,2,s)$	≥ 1137	$16 \leq s \leq 34$	$h(2,2,22)$	$= 21$	
$h(3,2,6)$	$= 15$		$h(2,2,24)$	$= 20$	
$h(3,2,8)$	$= 10$		$g(1,1,0)$	$= 60$	
$h(3,2,11)$	$= 5$		$g(1,1,s)$	$= 80 + 4s$	$1 \leq s \leq 5$
$h(3,2,13)$	$= 0$		$g(1,1,s)$	≥ 1204	$6 \leq s \leq 36$
$g(2,1,0)$	$= 22$		$h(1,1,4)$	$= 47$	
$g(2,1,1)$	$= 23$		$h(1,1,10)$	$= 53$	
$g(2,1,2)$	$= 34$		$h(1,1,15)$	$= 33$	
$g(2,1,3)$	$= 37$		$h(1,1,16)$	$= 58$	
$g(2,1,4)$	$= 40$		$h(1,1,18)$	$= 65$	
$g(2,1,s)$	$= 33 + 3s$	$5 \leq s \leq 9$	$h(1,1,21)$	$= 38$	
$g(2,1,s)$	≥ 1163	$10 \leq s \leq 34$	$h(1,1,23)$	$= 45$	
$h(2,1,6)$	$= 16$		$h(1,1,26)$	$= 49$	
$h(2,1,12)$	$= 15$		$h(1,1,28)$	$= 50$	
$h(2,1,14)$	$= 20$		$g(0,1,0)$	$= 60$	

Table 1: Computational results for the example of Figure 1.

[3]Note that the computational burden is surprisingly large for such a small problem; the number of g and h-values that were ultimately generated by our computer program, amounts to 471.

5 Computational Results

We have implemented the dynamic programming algorithm in the programming language C++ on a DEC 2100 A500MP workstation with 128Mb internal memory. All problem instances have the same cost structure as the one proposed by Cho and Shaw [11] (cf. Section 2). The first set of instances we consider is made publicly available by Cho and Shaw. The number of nodes in the tree varies from 5 to 30, whereas the maximum concentrator capacity is in the range of 41 to 951. For two instances (`exp15a` and `exp20b`) we found an optimal value which was lower than the value presented in Cho and Shaw; this is probably due to typing errors, since they also solved the problems with CPLEX. The results in Table 2 indicate that our algorithm is significantly faster than the algorithm presented by Cho and Shaw; 43 times as fast on average over all 11 problems, and even up to 76 times as fast on average over the 5 largest problems.[4]

problem	n	m	B	value	CPU sec.
`exp5a`	5	1	41	559	0.017
`exp9a`	9	1	51	1305	0.020
`exp9b`	9	1	70	559	0.020
`exp9c`	9	1	340	25847	0.023
`exp15a`	15	3	451	55566	0.028
`exp15b`	15	3	474	65693	0.039
`exp20a`	20	1	140	23382	0.029
`exp20b`	20	3	443	62859	0.038
`exp20c`	20	3	951	162936	0.059
`exp30a`	30	1	340	93238	0.044
`exp30b`	30	3	451	86717	0.077

Table 2: Computational results for the instances generated by Cho and Shaw.

We also tested 10 larger instances of this type using the same generator as Cho and Shaw (available on Shaw's Web site), with number of nodes varying from 50 to 200, and with the maximum concentrator capacity in the range from 100 to 1500. Our results for these instances are listed in Table 3, which indicate that larger instances can still be solved efficiently: instances with small concentrator capacities are solved within a second, whereas the running time for larger concentrator capacities is within minutes.

All of the above instances consisted of trees with an unbalanced structure as the tree in Figure 2. Due to this structure, the number of coefficients r for which $h(v, i, r)$ is feasible for the (v, i)-pairs with $v \in \{0, 16, 31, 35, 45\}$ is very large, which makes this type of instances relatively hard to solve. The real-life problem instances considered by Balakrishnan, Magnanti and Wong [5] are based on more balanced trees. These authors were not able to solve their largest problem (41 nodes) to optimality. Unfortunately, these instances are not publicly available. In order to compare our algorithm to their method, we slightly modified Cho and Shaw's problem generator so as to obtain more balanced trees with comparable node demands and edge capacities. We

[4] This may partly be due to differences in hardware; Cho and Shaw used a SUN SPARC 1000 workstation.

problem	n	m	B	value	CPU sec.
lanep50a	50	3	185	247768	0.058
lanep50b	50	3	392	197222	0.363
lanep50c	50	3	928	127577	3.311
lanep100a	100	3	190	632303	0.181
lanep100b	100	3	280	443484	0.539
lanep100c	100	3	772	342672	10.045
lanep200a	200	3	137	1453570	0.193
lanep200b	200	3	192	1417313	0.450
lanep200c	200	3	680	795365	13.992
lanep200d	200	3	1424	574739	80.648

Table 3: Computational results for instances with 50 to 200 nodes.

generated trees consisting of 25 up to 1000 nodes, and concentrator capacities ranging from 3,000 to over 10,000. For each of the problem sizes, we generated five instances with 3 concentrator types. The best, average and worst CPU times are reported in Table 4. The results indicate

n	$\min B$	$\max B$	CPU sec.		
			best	average	worst
25	4408	4869	0.339	0.507	0.665
50	4169	4763	0.697	0.934	1.248
100	3761	5360	1.161	1.795	2.602
200	7708	10218	31.535	71.680	127.619
500	3800	4762	8.335	11.407	18.209
1000	2907	3597	9.873	13.423	16.184

Table 4: Computational results for balanced instances with 25 up to 1000 nodes.

that for problem instances comparable to the ones of Balakrishnan, Magnanti and Wong, we obtain optimal solutions within a second, and for significantly larger instances the CPU time is still within minutes. An implementation of the algorithm as well as the modified generator with all of the aforementioned instances are publicly available on World Wide Web or by e-mail to one of the authors.

6 Concluding Remarks

In this paper, we have described a pseudo-polynomial time dynamic programming algorithm for the LANEP. Our model follows a bottom-to-top approach using two parametrized families of related subproblems. We consider general cost structures which enables the model to encompass many related situations without altering the (complexity of the) algorithm. Furthermore, we have given a formal proof of the correctness of the algorithm.

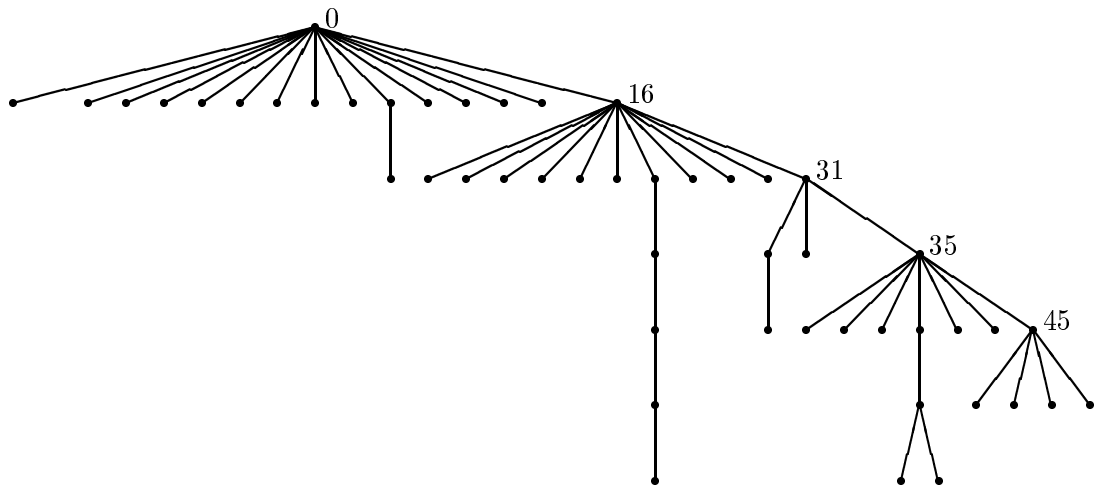


Figure 2: The tree of problem `lanep50a`

The computational experiments indicate that our algorithm is highly efficient. For problem sizes mentioned in the literature, which could not be solved to optimality within reasonable time, our algorithm finds an optimal solution within a second. Significantly larger problems, with many more nodes and higher concentrator capacities, can still be solved within seconds to minutes.

For the cost structure described in Section 2, Cho and Shaw have also studied LANEP. Although their algorithm is incorrect in the general case, they also consider the special case in which all existing edge capacities are zero. This special case is refered to as the Local Access Network Design Problem (LANDP). The typical characteristic of the LANDP is that for each edge on which demand is routed, fixed expansion costs are necessarily incurred. By including the concentrator location w into the state space, one knows that on each edge on the path from v to w capacity must be expanded. If an extra load from outside the tree must home on w, the extra costs on this path are equal to the variable costs on this path. Because of this exact (linear) relation between the costs of different loads coming into the tree, this parameter can be excluded from the state space. By applying a "left-to-right" instead of a "bottom-to-top" approach, they are able to solve LANDP in $\mathcal{O}(n^2B)$. A possible combination of the ideas used in our paper with such a "left-to-right" approach is a topic for further research.

References

[1] R.K. Ahuja, J.L. Batra, S.K. Gupta, and A.P. Punnen. "Optimal expansion of capacitated transshipment networks". *European Journal of Operational Research*, 89(1):176–184, 1996.

[2] A. Balakrishnan, T.L. Magnanti, and P. Mirchandani. "A dual-based algorithm for multi-level network design". *Management Science*, 40(5):567–581, 1994.

[3] A. Balakrishnan, T.L. Magnanti, and P. Mirchandani. "Modeling and heuristic worst-case performance analysis of the two-level network design problem". *Management Science*, 40(7):846–867, 1994.

[4] A. Balakrishnan, T.L. Magnanti, A. Shulman, and R.T. Wong. "Models for planning capacity expansion in local access telecommunication networks". *Annals of Operations Research*, 33:239–284, 1991.

[5] A. Balakrishnan, T.L. Magnanti, and R.T. Wong. "A decomposition algorithm for local access telecommunications network expansion planning". *Operations Research*, 43(1):58–76, 1995.

[6] I. Barany, J. Edmonds, and L.A. Wolsey. "Pacing and covering a tree by subtrees". *Combinatorica*, 6:221–233, 1986.

[7] D. Bienstock. "Computational experience with an effective heuristic for some capacity expansion problems in local access networks". *Telecommunication Systems*, 1:379–400, 1993.

[8] D. Bienstock, S. Chopra, O. Günlük, and C.-Y. Tsai. "Minimum cost capacity installation for multicommodity network flows". Working paper, July 1995.

[9] D. Bienstock and O. Günlük. "Capacitated network design — polyhedral structure and computation". Working paper, June 1995.

[10] S-G. Chang and B. Gavish. "Lower bounding procedures for multiperiod telecommunications network expansion problems". *Operations research*, 43(1):43–57, 1995.

[11] G. Cho and D.X. Shaw. "Limited column generation for local access telecommunication network design - formulations, algorithms, and implementation". Working Paper, January 1995.

[12] B. Gavish. "Topological design of telecommunication networks: Local access design networks". *Annals of Operations Research*, 33:17–71, 1991.

[13] B. Gavish. "Topological design of computer communication networks - the overall design problem". *European Journal of Operational Research*, 58(2):149–172, 1992.

[14] L. Gouveia and J. Paixão. "Dynamic programming based heuristics for the topological design of local access networks". *Annals of Operations Research*, 33:305–327, 1991.

[15] M. Grötschel, C. Monma, and M. Stoer. "Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints". *Operations Research*, 40(7):309–330, 1992.

[16] M. Grötschel, C. Monma, and M. Stoer. "Polyhedral and computational investigations for designing communication networks with high survivability requirements". *Operations Research*, 43(6):1012–1024, 1995.

[17] J. Hellstrand, T. Larsson, and A. Migdalas. "A characterization of the uncapacitated network design polytope". *Operations Research Letters*, 12:159–163, 1992.

[18] C. Jack, S-R. Kai, and A. Shulman. "Design and implementation of an interactive optimization system for telephone network planning". *Operations Research*, 40:14–25, 1992.

[19] D.S. Johnson and K.A. Niemi. "On knapsacks, partitions and a new dynamic programming technique for trees". *Mathematics of Operations Research*, 8:1–14, 1983.

[20] T.L. Magnanti and R.T. Wong. "Network design and transportation planning: Models and algorithms". *Transportation Science*, 18:1–55, 1984.

[21] H. Pirkul and V. Nagarajan. "Locating concentrators in centralized computer networks". *Annals of Operations Research*, 36:247–262, 1992.

[22] N.G.F. Sancho. "A suboptimal solution to a hierarchial network design problem using dynamic programming". *European Journal of Operational Research*, 83(1):237–244, 1995.

[23] A. Shulman and R. Vachani. "An algorithm for capacity expansion of local access networks". In *IEEE Infocom'90*, San Francisco, California, 1990.